Manual

Welcome to Promod's manual. Here you will find all the information you need to run this software. This program is a project for the Structural Bioinformatics subject and the Introduction to Python project of the Msc in Bioinformatics of Universitat Pompeu Fabra.

What is Promod?

Promod is a python tool whose goal is to form macrocomplexes of molecules starting from the interacting pairs of chains that will form the complex. It is compatible with protein chains, single and double DNA strands and RNA strands. It has several parameters available to play with according to the users needs.

In essence, the builder approach is similar to a genomic assembler: seeks for similar chains in the different PDB files given as an input and overlaps them. After that, joins the chains in a single model. Pair by pair the builder puts all the possible chains inside a single model

Background and scientific explanation

Introduction

Proteins are the executive molecules in all organisms. They perform a wide variety of functions, from small compounds transport to signals transduction or immune responses. However, proteins do not usually work individually in cells. Most proteins interact with other proteins (protein-protein interactions, PPIs) or molecules (DNA, hormones, drugs, among others), which are essential for a proper development of their activities. That is the reason why collecting PPIs can lead to a better understanding of protein functions, biological pathways and mechanisms of disease. The identification of such PPIs has been a challenging task for years. Experimental methods provide irrefutable evidence to test interactions between proteins, but they are expensive and time-consuming. Nowadays, computational approaches are being developed to reduce as much as possible the necessity of experimental data. Some advances have been made in this field, but the range of successful organisms is short and general frameworks are lacking at the moment.

Different experimental approaches for the identification of individual PPIs are available. The most accurate methods that allow the determination of the exact atom coordinates in the complexes are X-ray crystallography, Nuclear Magnetic Resonance and Electron Microscopy. These techniques are difficult to perform, they need very specialised equipments and a strong data analysis procedure. There are also high-throughput methods based on biochemical properties, such as Tandem Affinity Purification, which can detect multiple PPIs at once, but a high rate of false positive results can be expected and the information is not very precise. Protein microarrays are also being used as a screening method that can be easily automated and parallelised. Other traditional techniques, namely co-immunoprecipitation, yeast two-hybrid or pull-down, are still being used as well. All these approaches provide complementary views of PPIs and have their advantages and problems. The main one is the cost-effectiveness of the experiments, making in silico approximations more feasible and adequate for understanding PPIs at the atomic level.

Regarding the structural properties of the proteins, there are three main categories of methods for computational modelling of PPIs: homology or template-based modelling, *ab initio* or template-free modelling, and hybrid or integrative modelling. Homology modelling is based on the fact that the evolutionary information in both the sequences and the structures is important for PPI prediction. The latter are preferred, as most existing predictors use surface patch data, and only the residues in the interface have to be analysed, instead of the whole sequence of aminoacids. A pitfall of this method is that not all the 3D structures are available, although public databases such as PDB are unstoppably increasing in size. Template-free modelling needs more computational resources, as it explores all the possible orientation between the interacting molecules. The combination of prior knowledge about the individual structures of the components may help reduce the searching space, but still this approach is more challenging. This kind of techniques also require a careful

evaluation of the results by various means and refinement of the candidate models using biological information. Finally, integrative modelling combines experimental data and bioinformatics developments to narrow the possible complexes and save computational resources and time.

In this work, our aim is to develop a software that builds protein macrocomplexes using as input pairs of protein-protein interactions. Interaction with other molecules, such as nucleic acids, is also considered. To do this, our program is based on homology modelling. Therefore, the homology of the different chains is analysed, tridimensional structures are superimposed and energy levels of the final models are considered to propose the best possible solution. Another option would have been template-free modelling, but the required computational resources and the steps of evaluation and refinement excluded this possibility, because of the limited means and knowledge that our team has got.

Here we propose Promod as a first step approach to protein-protein and protein-nucleic acid complex modelling. It is distributed as both a Python package and a standalone application to be run in UNIX-based systems. Several parameters can be tuned to adjust to the user's needs. The final result is a single model with the best option according to the algorithm that is explained below.

The Promod method

The algorithm used by our program can be divided in three main steps. The first one is the analysis of the homology of the subunits. Then, the superimposition of the homologous structures is performed. Finally, energy levels in the models are taken into account to discard unlikely complexes.

Homology of the subunits In order to build the model, the program begins with several files, each of them containing information about two interacting chains, whether peptidic, DNA or RNA. We can overlap similar proteins from two different PDB files to check if two chains in those files are alike. If they effectively are, they can be joined in the same model. This is known as protein superimposition, and the whole process can be compared to a DNA sequence assembly. The main objective is to recognise that two sequences are the same and put them together in the correct spot.

Two chains in different PDB files must be homologous to be considered as part of the same protein and overlap them. To check if two proteins are homologous or not, a pairwise alignment is performed. This alignment consists of calculating the similarity between two sequences, taking into account both mutations and gaps, and then returning a score between 0 and 1. A score close to 1 means that both sequences are identical, therefore high score values are evidence of a large proportion of sequence identity. Only the homologous proteins will be overlapped and used to build the model later.

Superimposition of the 3D structure Two homologous sequences will probably have similar structures. However, it is also possible that two proteins with lower identity score may have similar structures in the tridimensional space, as a result of convergent evolution. Therefore, when two proteins are significantly different in sequence but they have similar structures, the may have the same role in our model and it is desirable to consider them. For example, two homologous proteins from distant species whose alignment do not pass our homology threshold, but they still preserve the structure.

In these cases, to test if two proteins really have similar structures, we need a measurement, such as the root median square deviation (RMSD). First, to calculate this value, we have to superimpose these two proteins. That means placing the atoms of both proteins in the same coordinates and orientation, to check how well they overlap in the space. Similar structures will have atoms in almost the same positions, while different structures will be more distant. This similarity between the superimposed proteins can be used to calculate the RMSD, which is the average distance between the superimposed atoms in the chains. A value close to zero indicates a perfect fit of the structures. RMSD will increase when the differences between the protein structures increase.

Due to all this, RMSD must also be a filter to account for those structures that are different, even when we had considered them homologous in previous steps.

Analysis of the energy levels Lastly, it is important to consider the final energy levels of the complex. A good model should have minimum energy, as functional, naturally occurring complexes are the ones with the lowest energy among the set of possible foldings. Situations such as two atoms very close to each other, aminoacids with hydrophobic residues located in the external part of the macromolecule and several other cases can increase the final energy of the complex, which should be then corrected.

After that, it is possible that the position of the atoms inside the model is not the most adequate. Sometimes, despite having evidence of interaction between two chains, collisions or clashes appear when they are joined in the structure. Taking this into account avoids impossible models, such as those with chains crossing with each other, which will be thermodynamically unstable and unlikely to happen.

Features

The most striking features of Promod are:

1. Building of macromolecular complexes from basic input data (pairs of interactions between molecules).
2. Optimization of the final model using MODELLER.
3. Graphical user interface (GUI), with the same functionalities as the command line interface (CLI).

Implementation

The Promod package is implemented in Python3. It strongly depends on the Biopython library, which is an installation requirement for the correct execution of this software. The GUI has been developed under the Tkinter framework, providing a user-friendly interface and, thus, avoiding the use of the command line. The implementation of the GUI and the CLI are independent, so each one can be executed on its own. This software works with well-established bioinformatics formats, such as FASTA and PDB files, so no atypical formatting of input data shall be conducted. Additionally, two scripts are provided to divide a given PDB file in separate pairs of chains and join them if there is an interaction between the molecules. Some examples are also included for the testing of the program. Promod is freely available from the following Github repository. URL: https://github.com/Fabian-RY/SBI-Python-project.

The formatted documentation of the package includes dependencies and installation instructions, examples of use and a full tutorial with sample code. Each function has got its own documentation that instructs on the particularities when importing them to be used independently. The users could learn all knowledge of the library by looking up the detailed documentation. The main functionalities are further explained in the tutorial section.

Discussion

Promod is a basic tool that relies on information from both the sequence and the tridimensional structure of pairs of interacting molecules to build a final complex model. As we can see in the analysed examples, it is successful with small complexes and it can even handle nucleic acids interaction with proteins. The MODELLER final step adds an additional refinement of the proposed model, returning to the user a good quality structure. All this methodology is built in an easy-to-use package, well documented and with a GUI to save unpleasant command line work for the standard user.

However, the main objective of modelling software is to provide new knowledge without the need of huge amount of experimental data, such as the determination of all paired interactions, in this case. A limitation of our program is the necessity of input protein-protein or protein-nucleic acid interaction. This type of software is being broadly developed by means of various approaches, which are proving to be very successful. Both docking and homology modelling paradigms are being applied to achieve this goal.

One of the strategies being exploited is evidence combining methods. The core of this strategy is integrating evidence from multiple sources, including them in comprehensive databases for later integration. They are called gold standard databases and they contain information for both training and testing of the new methods. The annotation of paired protein interactions goes beyond structural features. For instance, evolutionary relationships, functional features, network topologies, sequence-based signatures, structure-based signatures, and text mining information is recorded in these databases. Finally, machine learning algorithms are fed with subsets of data and performance is measured to propose the better candidates, depending on target species, data sources, demand of accuracy and coverage. These evidence combining methods are performed repeatedly to find converging results with different input data and classifiers.

As we can see, template-based methods are leading the in silico modelling techniques. It is reasonable to believe that there are a limited number of possible interactions and that, once we have big enough databases and curated interaction catalogs, most of the PPIs should be easy to model with high confidence. However, *ab initio* modelling has also yielded promising results, mainly from competitions such as CASP, CAPRI or CAMEO, where world-class groups bring their latest developments to test their performance with real problems. Of course, these solutions are computationally expensive and require a long time to return a final model. In addition, complexes with weak interactions where the conformational state changes upon binding are a big challenge for docking software.

A large number of information is nowadays available from high throughput experimental techniques and a lot of structural bioinformatics software has been developed to integrate these data. The best performance of in silico techniques has been achieved at the tertiary structure level of proteins. Nevertheless, quaternary structures are the ones responsible for the majority of biological functions and their knowledge is essential to disentangle protein interaction networks in both physiological and pathological scenarios. It is clear that hybrid approaches, joining atomic-level experimental structures, database information and the newest machine learning techniques, will give promising results in the near future.

Future perspectives

As we have already mentioned, there are diverse strategies for modelling PPIs. The development of refined algorithms to perform this task is far beyond our current knowledge in structural bioinformatics. Therefore, if we had time and resources to expand our project, we would focus on improving the user experience and the accessibility to existing data.

First, we would like to implement different ways of modelling. It would be useful for the user to choose whether to use a template-based or a template-free approach. We could also use existing Biopython packages to perform certain operations, but further research and testing would be needed to decide the better candidates.

Next, the result of the modelling operations could return automatic reports about the process. Not only the final model would be reported, with the tridimensional structure and their characteristics, but also information about the reasons why our software has discarded certain possible models. Therefore, direct visualization of these other options and energy plots for user reference should be displayed. A good option for the format of this report would be a Jupyter notebook, so the user can interactively check all the available resources.

Another functionality that could be feasible to achieve is the automatic download of structures from public databases, such as PDB. Both sequences and structures can be obtained using Biopython modules and some keyword and ID search should be easy to implement, as well. These would be the input data for the additional scripts that build the pairs of interacting molecules to run the core Promod builder.

Finally, it would be ideal to develop integration options of biological information to help build better complexes, in line with trending research in the field. We are not aware of Python packages that would allow to directly implement this kind of work, but maybe external tools are capable of doing it. However, a broad review of the latest literature and a challenging programming development would be needed to know the most promising approximations.

Conclusion

Although computational approaches for building macromolecular complexes are far to be perfect at the moment, a lot of effort is being made to develop strategies to overcome the pitfalls in this field. We have provided a software that, despite not using any novel strategy, achieves notable results when using already defined structures. Further steps using machine learning approaches and a broader range of training data would improve the performance and confidence of this type of programs. Finally, the integration of different types of biological data will also be a key step in the progress of in silico modelling of protein-protein interactions. The achievement of better methodologies will definitely have an impact in applied fields, such as drug discovery, biomedical research or food industry.

Input

The input is a folder with two or more files: each file contains two proteins, a protein and DNA/RNA strand or 2 single DNA strands (which may or may not form a double strand), which represent an interaction between those chains. Two structures are considered to be interacting if the minimum distance between them is in a range of a few Argmstrongs (1-10) . However, in lesser distances, forces between atoms are strong enough to produce changes between the chains and force them to adopt a different conformation, an thus, a realistic model must keep the residues at an adequate distance to consider it correct.

Output

The output is mainly one pdb file with all the possible chains joined in the selected folder. However, if the optimized option was selected, several pdb files will be created in the same directory. These are different approaches made by modeller to optimize the energies of the model and the files it used. The model will be saved as final_model.pdb

If the model is required with minimum energy, and thus the -optimize flag is used, then several files will be saved. Modeller will save some mid-step pdbs as 'final-model.DXXXXX.pdb' and the fully optimized will be saved as 'optimized.pdb'

Tutorial

Installing dependencies

In order to make Promod work, there are some dependencies that must be installed before executing the software: python3 and Biopython

If you're using Windows, you can download python3 from its website. If you got Linux, there's high chance that you already got it installed.

Independently of the operating sistem you got, you can install biopython using pip.

```
{sh, eval=F} pip3 install biopython
```

Instalation

The recommended way to install promod is by using pip, exaclty with the same command as we did with biopython previously `{sh, eval=F} pip3 install promod`

You can install Promod easily by downloading it from the github repository and running the next command in the downloaded folder. This is, for now, the recommended install way

```
{sh, eval=F} pip3 install .
```

Alternatively, you can also run the setup script installation commands. `{sh, eval=F} python3 setup.py install`

Hands on: how to use promod

Promod has a command line interface which explains briefly how to use it before executing it.

`promod -h`

There are 3 mandatory commands while the rest are optional. The mandatory ones are related to the input files and output folder, necessary for the program to work, and they are *-i, the input folder; -o, the output folder; and -f, the fasta file of the sequences.* All of them will be covered in this manual.

It also has a graphical interface wich accepts the same paramenters but graphically with message boxes and graphical stuff. It's an independent executable, so the CLI one can be used for automation without confusion. The comand is simple:

`{sh eval=F} promod-tk`

Paramenters

Input folder The input folder should contain, at least, 2 pdb files. This folder may contain other files or subfolders; however, the program will ignore other files and will not check subfolders to find more pdb files. If you want to include some pdb, include it in the folder before running the program.

Beyond that last thing, no more things are necessary to know about the input folder. However, it's a mandatory argument, and should be indicated with -i or –input-folder tags

Output folder The output folder is the folder where the output files are written. This is mandatory, and no special folder is required. Just a warning, as, for the moment, the model file is written as 'final_model.pdb' and thus any other file with that filename will be overwritten.

The output folder is indicated with the -o or –output-folder tags

Distance You can indicate the minimum distance to consider that two proteins do not clash. The model will discard interactions with too many atoms at lesser distance than the value indicated in Argmstrongs. Being too strictive can discard some interactions, but being too flexible can force chains to overlap and not being energetically favorable.

You can indicate the distance with the -d or –distance tags

Threshold The threshold is the minimum score alignments must reach in order to insert the chain in the model and determinate its homologous protein. Usually a very high value (0.9 or higher) is recommended to ensure that the chains are correctly identified. However, for sequences with less homology might be needed to reduce this value.

Fasta file The fasta file contains the sequences of the chains and the id for the stoichiometry. And it's a critical file, as the sequences must be homologous for the chain in the pdb. We can hold up to 5% of differences (by default, using -t parameter can modify this threshold). However, if one of the chains (if no stoichiometry indicated) or one of the chains in the indicated stoichiometry differs too much, the program will exit. This is use to avoid guessing posible chains that could match as that would produce unexpected results.

Note: That means the sequences in the fasta should be of a similar length than equivlent chain in the pdb. For example, the fasta file and the pdbs directly downloaded from Protein Data Bank may differ in the initial and ending residues, as they are quite difficult to model, and usually are removed from PDBs.

Stoichiometry file An additional file with the stoichiometry can be given to the program to make sure that the final protein will contain no more than the indicated chains. This file must be properly formatted as follows:

chain_id_in_fasta,number_of_columns

one per line

Let's see the stoichiometry file of the example 1

```
3e0d_A,2
3e0d_B,2
3e0d_C,2
```

Starting pdb Promod's result is highly dependant on the first pdb used to build the model. In fact, results can be very different by varing the starting pdb. Thus we provide an optional argument to select this staring pdb to select different starting pdbs. This way, the same pdb with the same parameters will provide the same model. By default, the pdb files are sortered alphabetically and the first one in this sorted list is chosen.

The starting pdb can be selected with -start

Uninstalling

You can uninstall everything by using if installed using pip3

```
{sh, eval=F} pip3 unistall promod
```

However, if installed calling setup.py, files must be deleted manually.

Examples

Example 1 (3e0d)

3e0d is a small complex formed by 2 protein chains and 2 double DNA strands. It's a subunit of the eubacterial DNA polimerase but it's perfect for an initial testing of our project, so we can test how the program works, the time it takes and if the result is similar to the original one. In this cases, we have 6 different interactions, as each DNA strand is counted as a single one interacting with another one and binded to the protein chain by one of them. In fact, this complex can be thought as a dimer: two monomers formed by a protein and a double strand DNA, which interact by some residues in their protein chains.

So, the first test to our program consists on joining the different pdb files into one single structure, without further information about the stoichiometry. Therefore, the program will try to build the protein with only the information provided by the pdb files and the fasta file. The command to build it is the next one:

```
{sh, eval=F} promod -i examples/example_1/chains/pairs -o examples/example_1/results  \
-f examples/example_1/3e0d.fa
```

We just need to give the folder with the input pdbs (-i), the desired output folder (-o) and the fasta file with the sequences of the proteins. The program will take the different pdbs on the folder (By alphabetical order, to make it reproducible) And the result it's incomplete:

There we can see that there is a missing double strand DNA. So, let's see how we can complete the model. We can see that there is a missing protein, let's force the program to include it. We have two different ways of

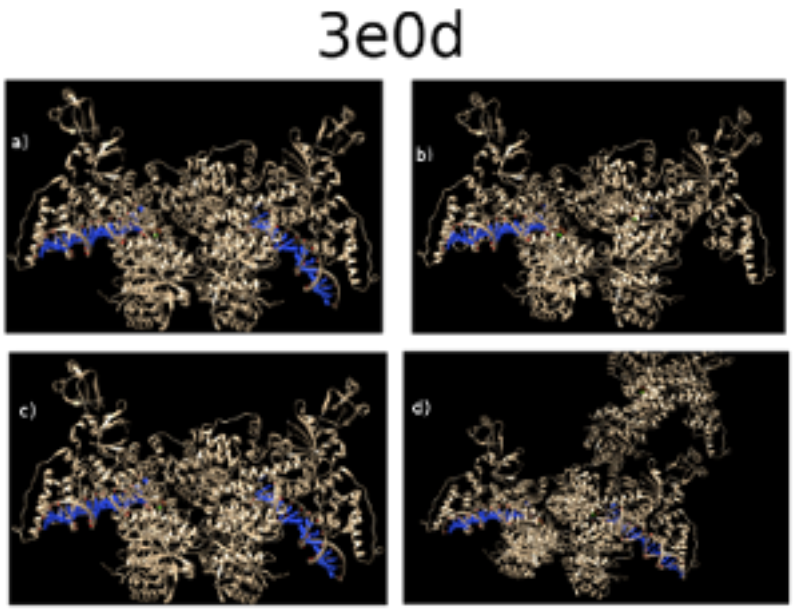

Figure 1: 3e0d structure obtained from: a) rcsb.com b) promod without any extra info c) promod with starting point and selected stoichiometry d) without stoichiometry control

indicating this: The easiest one is selecting the starting complex of our model (which contains those missing parts). This results that different starting points can result in different models, so, it's important to select a good one. By default, the pdbs filenames are sortered alphabetically and the first one is chosen as the starting point. In fact, this is an important choice: the differences between not chosing one (Figure 1.b) chosing an adequate one (Figure 1.c) or chosing another one (Figure 1.d) can result in models with extra chains or with less chains than expected

The other not-so-difficult way is to indicate the stoichiometry of the complex. In this case we will focus in selecting the starting pdb, so the command would be.

```
{sh, eval=F} promod -i examples/example_1/chains/pairs -o examples/example_1/results \
-f examples/example_1/3e0d.fa -start examples/example_1/chains/pairs/3e0d_ZG.pdb
```

Yay! This is fare more similar to the original structure, very close to the original model. However, this is a very simple protein, and bigger complexes may are harder to build. However, from this example, we learnt that:

- The program can build a model without any indication. However, it might be incomplete o have extra chains.
- The starting pdb is a important choice that can influence the final model. Thus it is important to remember which starting point gave which result. The same starting pdb will gave the same output.

Example 2 (6gmh)

6gmh is a very big complex formed by 24 different chains, with a double strand of DNA. There are several chains that interact between them, but there are not repeated sequences: each monomer is unique.

Thus, lets try to build it:

```
{sh, eval=F} promod -i examples/example_2/chains/pairs -o examples/example_2/results  \
-f examples/example_2/6gmh.fa
```

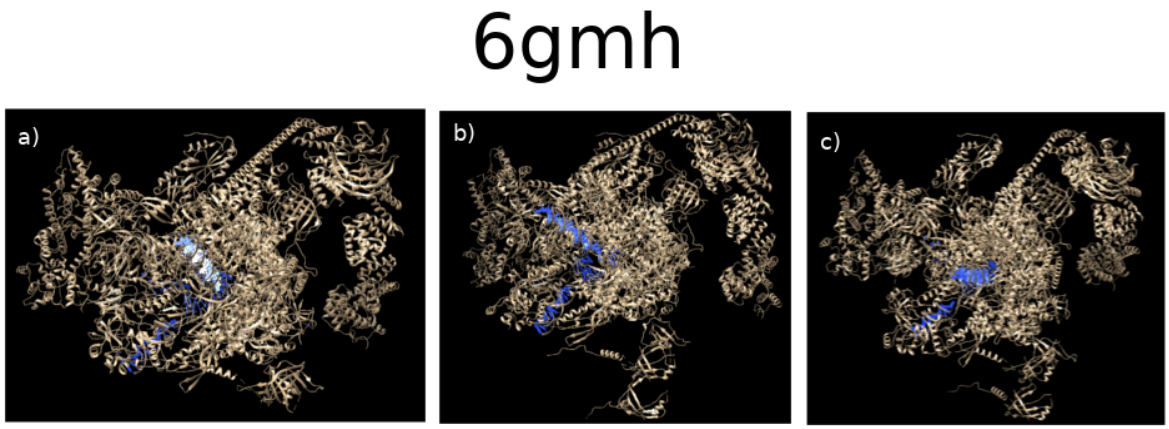

Figure 2: 6gmh structure obtained from a) rcsb.com b) builded without assistance c) selecting starting point and stoichiometry

In this case, there are several chains that hasn't been added to the model, and some are repeated. So, let's select a different starting point and limit the chains with the stoichiometry, so at maximun there is 1 copy of each chain.

```
{sh, eval=F} promod -i examples/example_2/chains/pairs -o examples/example_2/results  \
-f examples/example_2/6gmh.fa -start examples/example_2/chains/pairs/6ghm_VA.pdb \         -s
examples/example_2/6gmh.stoic
```

The 6gmh.stoic file contains the stoichiometry of the protein. In essence, is a csv file, without header, in which each line follows the same structure: ,. Each chain_name must be in the fasta as a sequence

This starting pdb was not in the model, and with this we force it to be included in the model, and start growing from there. And this way, the result contains far more chains in the model and it's far more similar to the original one.

Example 4 (5nss)

Here we want to show how to use two parameters we haven't shown yet: distance and threshold. Distance is a value that considers if two atoms collide, which is not allowed, as two atoms very close would have very high energy (if the separation between them is less than distance, it's considered invalid. However, to avoid some errors, up to 10 atoms colliding are allowed to consider a valid interaction). Threshold, however, makes reference to the homology percentage between the pdb sequences between themselves and between the sequences in the fasta. This allows for certain flexibility in sequences with some mutations that may alter the conformation of the chain.

```
{sh, eval=F} promod -i examples/example_3/chains/pairs -o examples/example_3/results  \
-f examples/example_3/5nss.fa -d 2 -t 0.9          -start examples/example_3/chains/pairs/5nss_NG.pdb
```

In this case, the result is quite good, but let's focus on the speed of the program. As the number of interactions grew, the time it takes for the program also increases. However, there are a few things to comment about it.

1.- Adding a new chain that passes all the checks is the most computationally expensive moment. The number of checks grows as the number of chains of the model increases.

2.- It takes less time to discard an interaction that has no homologous already in the model than adding a new chain. During the first steps of the program, until the model has grown enough, this is the most common

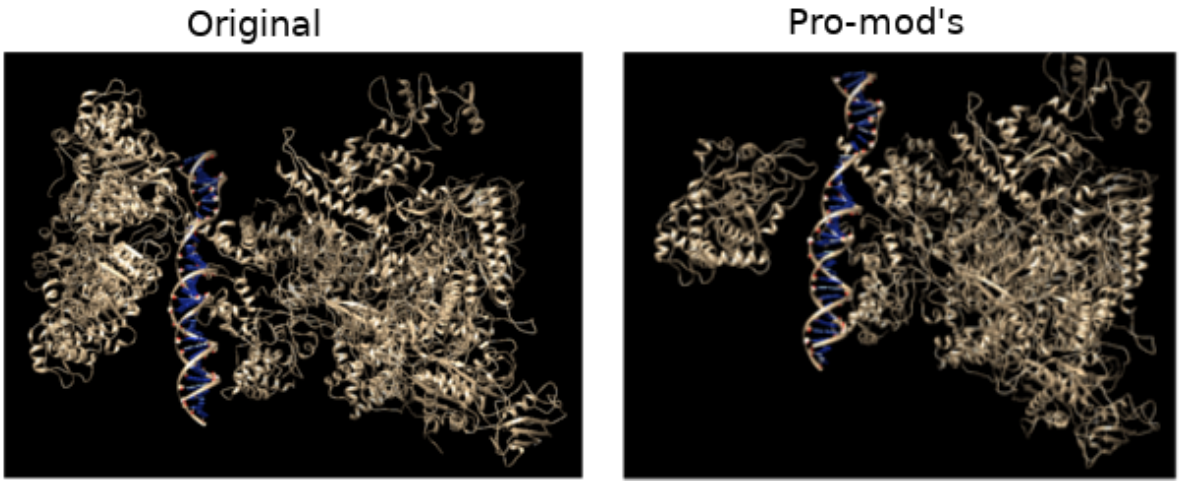

Figure 3: 5nss structure obtained from rcsb.org vs composed one

case. The number of alignments done is relatively low as there are a few chains in the model, but it can be an issue

3.- Once the model has a considerable number of chains, and particularly if there are several copies of a monomer, it's also noticeable that discarding an homologous protein because doesn't fit with the current parameters is a very time-consuming processes. The program tries to introduce the chain using all the possible alignments whose score is higher than the threshold. This is an important issue for proteins with a lot of copies of several monomers.

4.- The stoichiometry must be carefully selected to reach the desired structure. It's possible that the program cannot construct a model with the desire stoichiometry, but it's also possible that the stoichiometry wasn't what the user meant

Example 5 (6om3)

In this last example, let's talk about how the software behaves about the number of input pdbs and stoichiometry related stuff.

```
{sh, eval=F} promod -i examples/example_4/chains/pairs -o examples/example_4/results  \
-f examples/example_4/6om3.fa -d 0.3 -t 0.95 \         -s examples/example_4/stoic.csv
```

In this last example, let's talk about how the software behaves about the number of input pdbs and stoichiometry related stuff.

There are som chains left in the protein builded with promod, which can be added used custom values of distance or an apropiate starting point, but let's ignore that and focus on topics still not covered in the manual.

If you indicate a stoichiometry file but it doesn't exist, it exits. We could have changed it to a non-stoichiometry builder but we think it's better to do things explicitly as you may or may not notice this error.

An important thing about the stoichiometry is that is critical to make sure that all the model is builded as you want. For example, without any assistance, the work heres is quite good, but with a selected stoichiometry, the result is missing some chains. Here we want to say that is important to check the stoichiometry file to make sure the chains are correctly selected and the number of them is adequate to our purpose.

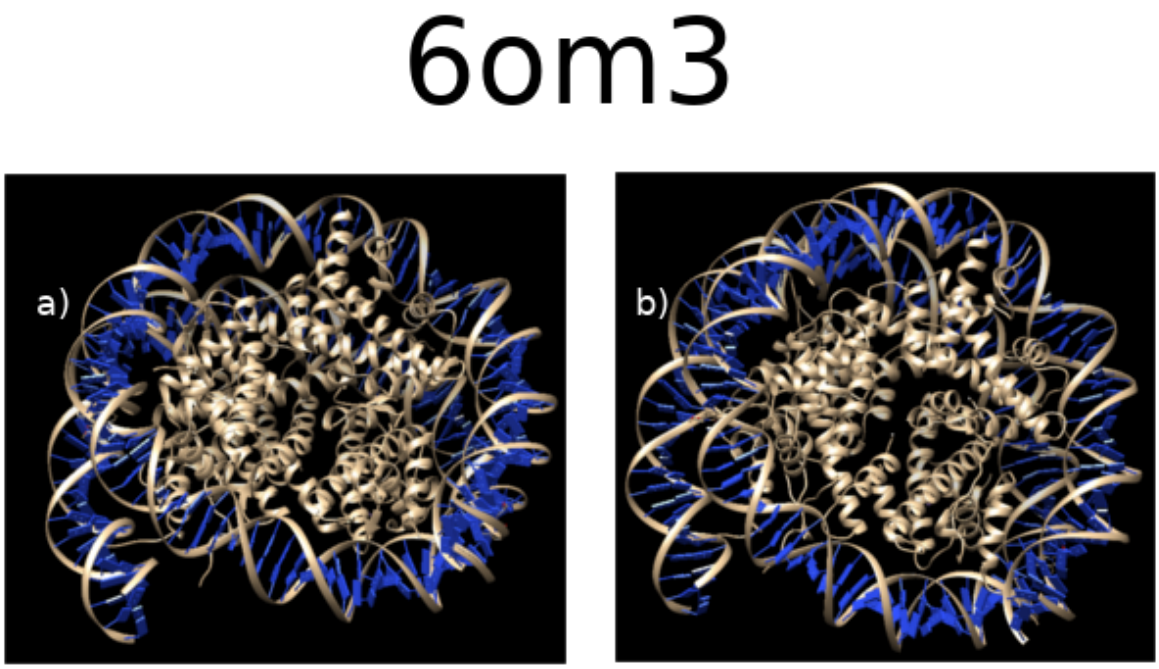

Figure 4: 6om3 builded from: a) rcsb.com b) promod with stoichiometry

Limitations

1. The sequences in the pdb and the sequences in the fasta must be similar. The program admits a certain degree of tolerance (which can be selected by the user using the -t parameter) but using sequences in the fasta which are fairly different from the pdb will not allow the assembly of the protein. This is particularly difficult for pdb in which protein tails are not well modelled and they're not present in the pdb but they're in the fasta. The pairwise alignment with the tail will drop the score of the alignment under the threshold, forcing to decrease it or even mistake to which sequence in the fasta corresponds to jus by chance. Thus, our recommendation is to prepare the fasta file with the sequence as similar as possible to avoid mistakes. In case you needed, in this package there is a script included, pdbsplit.py, which can give you the sequences of the chains inside the pdb file, avoiding this kind of errors up to ceratin degree. So, if you're looking for the effect of certain mutations in the desired protein, might be worthy prepare the fasta file according to the pdb sequences.
2. The running time of the program is proportional to the number of pdbs selected and is affected by the order of the structures and the starting point. The most critical steps: Reading all the pdbs sequences (However, the higher number of interactions, more chains might be added). This is a tradeoff between number of interactions and speed.
3. Selecting different starting points can produce different models. It's difficult to say which is the most adequate starting point, as may vary according to the goal of the assembly.
4. The fasta file require to be specially prepared for this application: There should not be repeated sequences (with at least an alignment score equal or higher than threshold selected). This also applies for sequences with unknown aminoacids, as they are marked as X in the fasta file and therefore will mismatch with the sequence in the pdb. Those sequences are usually missing in the final model

Bibliography

Chang, J., Zhou, Y., Qamar, M. T. U., *et al.* Prediction of protein–protein interactions by evidence combining methods. *International Journal of Molecular Sciences* **17**, 1946 (2016). doi: 10.3390/ijms17111946

Ding, Z., Kihara, D. Computational identification of protein-protein interactions in model plant proteomes. *Scientific Reports* **9**, 8740 (2019). doi: 10.1038/s41598-019-45072-8

Hayes, S., Malacrida, B. , Kiely, M., Kiely, P. A. Studying protein–protein interactions: progress, pitfalls and solutions. *Biochemical Society Transactions* **44**, 994-1004. doi: 10.1042/BST20160092

Liu, S., Liu, C., Deng, L. Machine learning approaches for protein–protein interaction hot spot prediction: progress and comparative assessment. *Molecules* **23**, 2535 (2018). doi: 10.3390/molecules23102535

Nealon, J. O., Philomina, L. S., McGuffin, L. J. Predictive and experimental approaches for elucidating protein–protein interactions and quaternary structures. *International Journal of Molecular Sciences* **18**, 2623 (2017). doi: 10.3390/ijms18122623

Sarkar, S., Gulati, K., Kairamkonda, M., *et al.* Elucidating protein-protein interactions through computational approaches and designing small molecule inhibitors against them for various diseases. *Current Topics in Medicinal Chemistry* **18**, 1-18 (2018). doi: 10.2174/1568026618666181025114903

Keskin, O. , Tuncbag, N. , Gursoy, A. Predicting protein-protein interactions from the molecular to the proteome level. *Chemical Reviews* **116**, 4884-4909 (2016). doi: 10.1021/acs.chemrev.5b00683